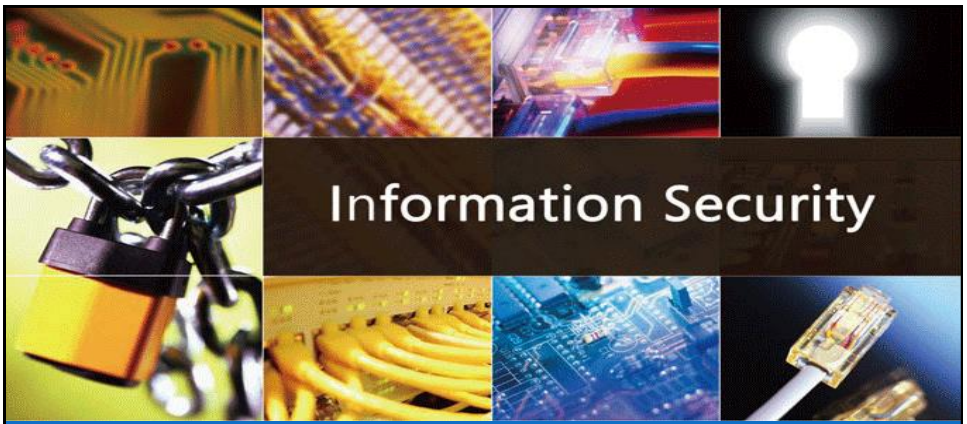


Database security

Lecturer: Nguyễn Thị Thanh Vân – FIT - HCMUTE

Objective

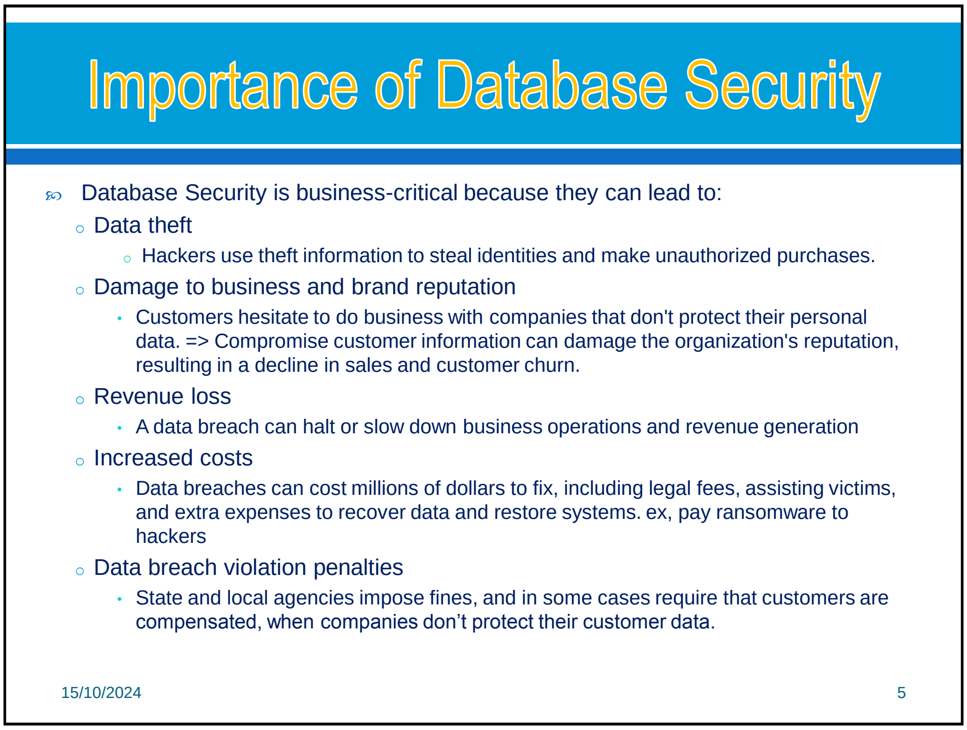
- Understand the importance of databases
- Understand the importance of securing data stored in databases
- Learn how the structured nature of data in databases impacts security mechanisms
- Understand attacks and defenses that specifically target databases

Importance of Database

- Databases Run Your Life. Databases are everywhere,
  - They effect on our daily lives is extensive, databases are responsible for many of the services we utilize daily.
  - They help human to search through millions of filing cabinets to find a particular record.
  - They are a standardized and performant way for programs to store, retrieve, and mutate data.
  - There are quite a few things that would be very difficult if not impossible without databases… The Internet, GPS, Electronic Banking, and much more.
  - They make everything easier, tons of other interesting and useful stuff.

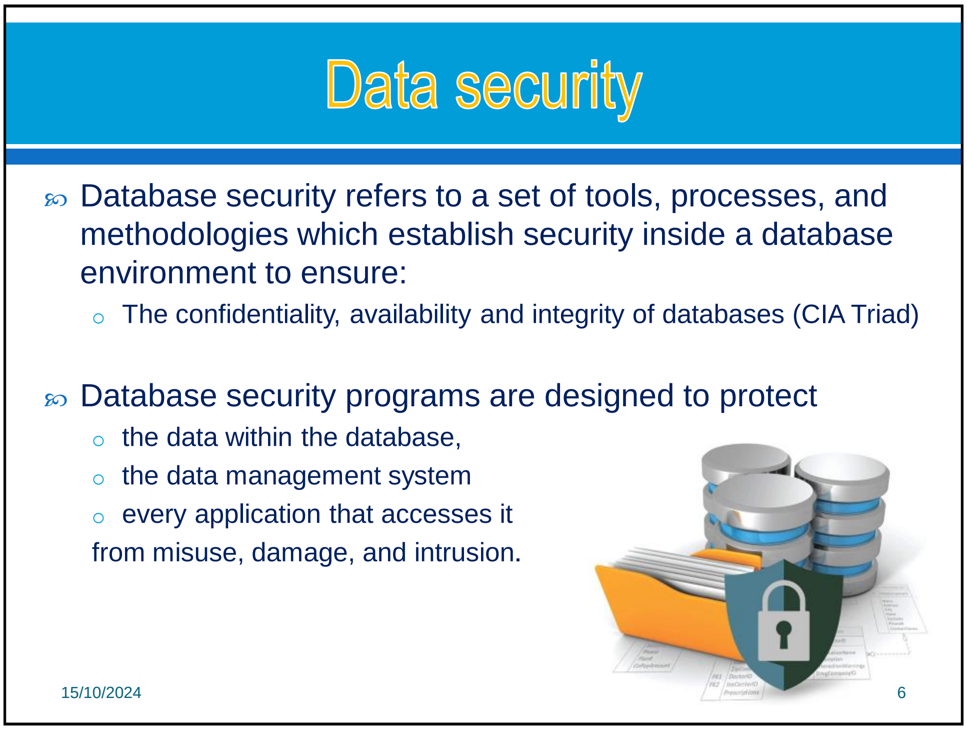
Characteristic of Database

- Databases store massive amounts of sensitive data
- Data has structure that influences how it is accessed
- Transactional nature of queries (updates or reads)
- Accessed via queries or programs written in languages like SQL
- Derived data or database views



Importance of Database Security

- Database Security is business-critical because they can lead to:
  - Data theft
    - Hackers use theft information to steal identities and make unauthorized purchases.
  - Damage to business and brand reputation
    - Customers hesitate to do business with companies that don't protect their personal data. => Compromise customer information can damage the organization's reputation, resulting in a decline in sales and customer churn.
  - Revenue loss
    - A data breach can halt or slow down business operations and revenue generation
  - Increased costs
    - Data breaches can cost millions of dollars to fix, including legal fees, assisting victims, and extra expenses to recover data and restore systems. ex, pay ransomware to hackers
  - Data breach violation penalties
    - State and local agencies impose fines, and in some cases require that customers are compensated, when companies don't protect their customer data.



Data security

- Database security refers to a set of tools, processes, and methodologies which establish security inside a database environment to ensure:
  - The confidentiality, availability and integrity of databases (CIA Triad)

- Database security programs are designed to protect
  - the data within the database,
  - the data management system
  - every application that accesses it

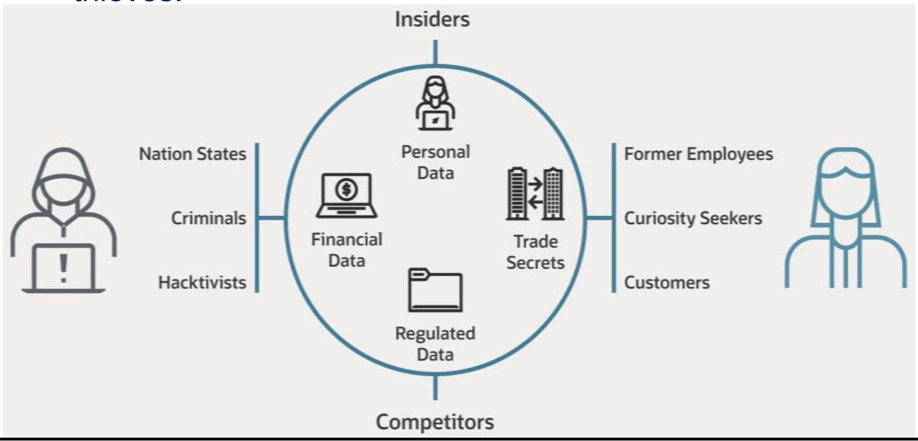
  from misuse, damage, and intrusion.

Database Security Issues

- Database security a broad area
  - Legal, ethical, policy, and system-related issues
- Types of threats to databases
  - Loss of integrity
    - Improper modification of information
  - Loss of availability
    - Legitimate user cannot access data objects
  - Loss of confidentiality
    - Unauthorized disclosure of confidential information

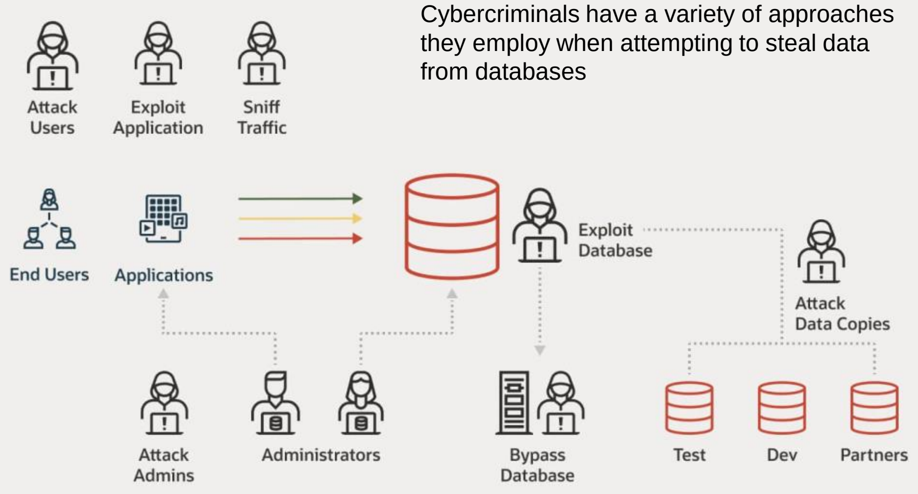
The challenges of database security

- Databases are valuable repositories of sensitive information, which makes them the primary target of data thieves.



The challenges of database security

Cybercriminals have a variety of approaches they employ when attempting to steal data from databases



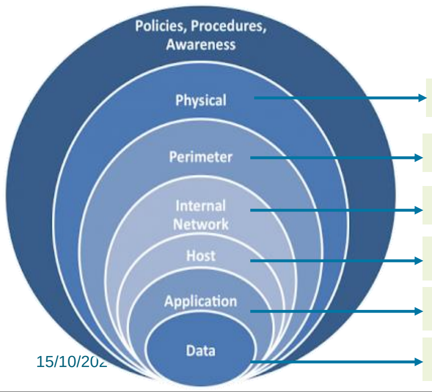
Database Security Threats

- Insider Threats
- Human Error
- Database Vulnerabilities
- SQL/NoSQL Injection Attacks
- Buffer Overflow Attacks
- Denial of Service (DoS/DDoS) Attacks
- Malware
- An Evolving IT Environment: trends can lead to new types of attacks on databases
  - Growing data volumes: must be highly scalable to address future reqs
  - Distributed infrastructure: ex cloud => security solutions more difficult
  - Increasingly tight regulatory requirements: become more challenging
  - Cybersecurity skills shortage: more difficult to defend critical DB

15/10/2024 10

Solution: Defense in depth

- a defense in depth (DiD) security strategy places multiple controls across the IT system.
  - If one layer of protection fails, then another is in place to immediately prevent the attack



Lock, Camera

UTM, Firewall, VPN, Routers,…

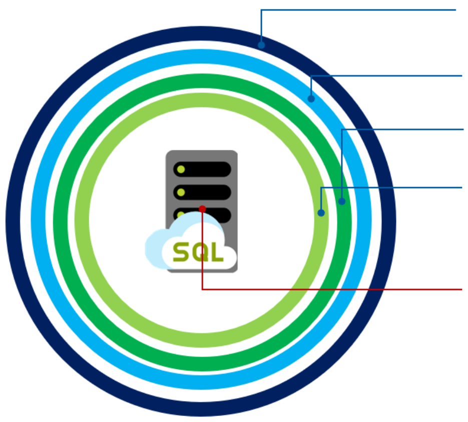
VLAN, IPS, IDS,…

OS, Update Management, Enpoint Security,…

Application Control, Antivirus,…

AAA, Encryption, Digital Signature…

Multiple layers of data protection



NETWORK SECURITY

ACCESS MANAGEMENT

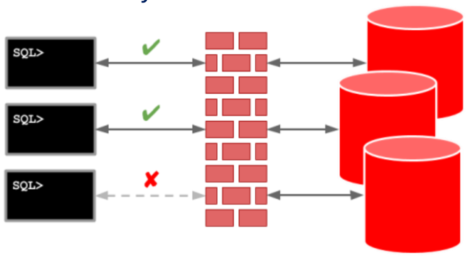
THREAT PROTECTION

INFORMATION PROTECTION

CUSTOMER DATA

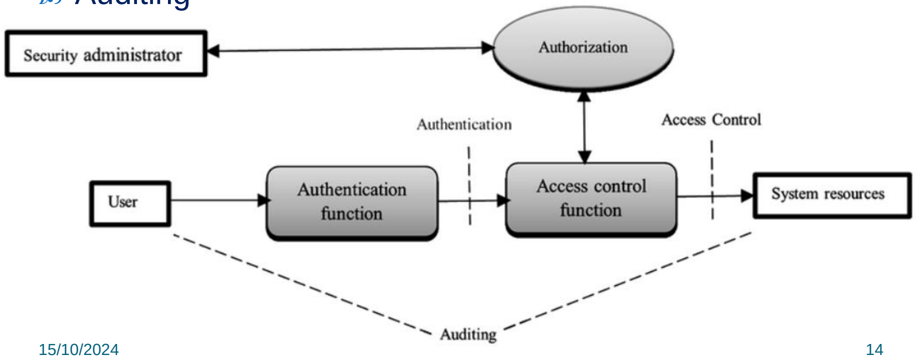
Layer 1: Network security

- Firewalls serve as the first line of defense in DiD database security.
  - a separator or restrictor of network traffic, which can be configured to enforce your organization's data security policy.
  - increase security at the OS level by providing a chokepoint where your security measures can be focused.
  - Make sure that your firewall is properly configured in such a way as to cover all security weaknesses.



Layer 2: Access management

- Authentication: DB, OS
- Authorization
- Access control
- Auditing



Authentication

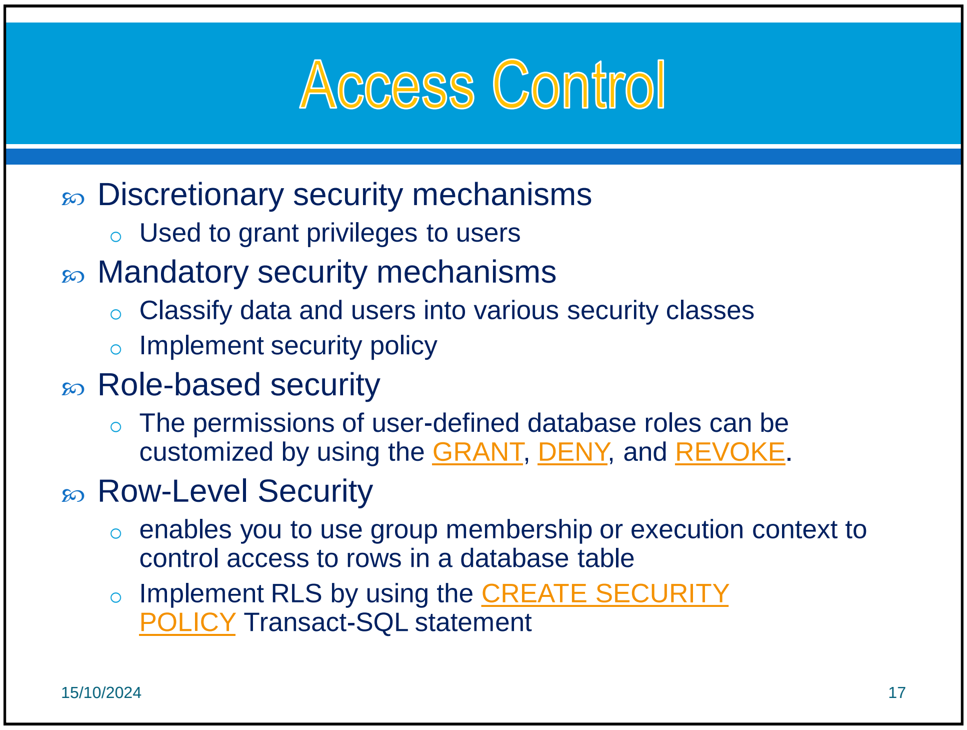
- Authentication: User must log in using assigned u/p
  - DB level, OS level
  - users and applications should use separate accounts to authenticate.
    - limits the permissions granted to users and applications
    - reduces the risks of malicious activity.
    - It's especially critical if application code is vulnerable to a SQL injection attack.
- Zero Trust is a framework for securing infrastructure and data for today's modern digital transformation.
  - securing remote workers,
  - hybrid cloud environments,
  - and ransomware threats

Slide 30- 15

Authentication, User, and DB Audits

- Authentication: User must log in using assigned u/p
  - DB level, OS level
  - users and applications should use separate accounts to authenticate. This limits the permissions granted to users and applications and reduces the risks of malicious activity. It's especially critical if application code is vulnerable to a SQL injection attack.
- Login session
  - Sequence of database operations by a certain user
  - Recorded in system log
- Database audit
  - Reviewing log to examine all accesses and operations applied during a certain time period

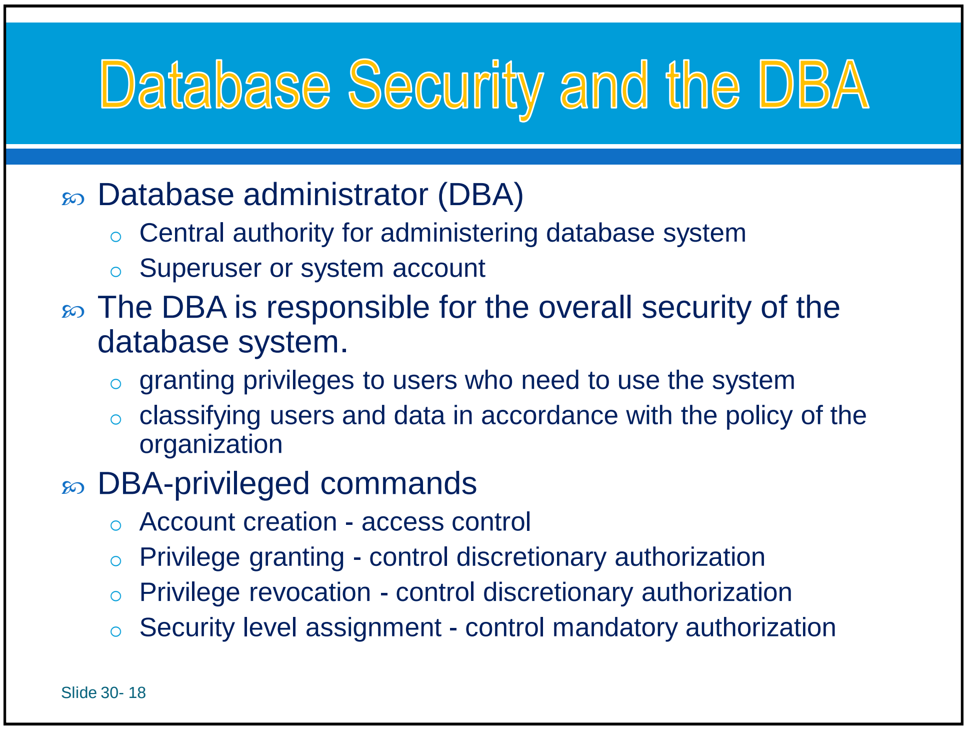
Slide 30- 16



Access Control

- Discretionary security mechanisms
  - Used to grant privileges to users
- Mandatory security mechanisms
  - Classify data and users into various security classes
  - Implement security policy
- Role-based security
  - The permissions of user-defined database roles can be customized by using the GRANT, DENY, and REVOKE.
- Row-Level Security
  - enables you to use group membership or execution context to control access to rows in a database table
  - Implement RLS by using the CREATE SECURITY POLICY Transact-SQL statement

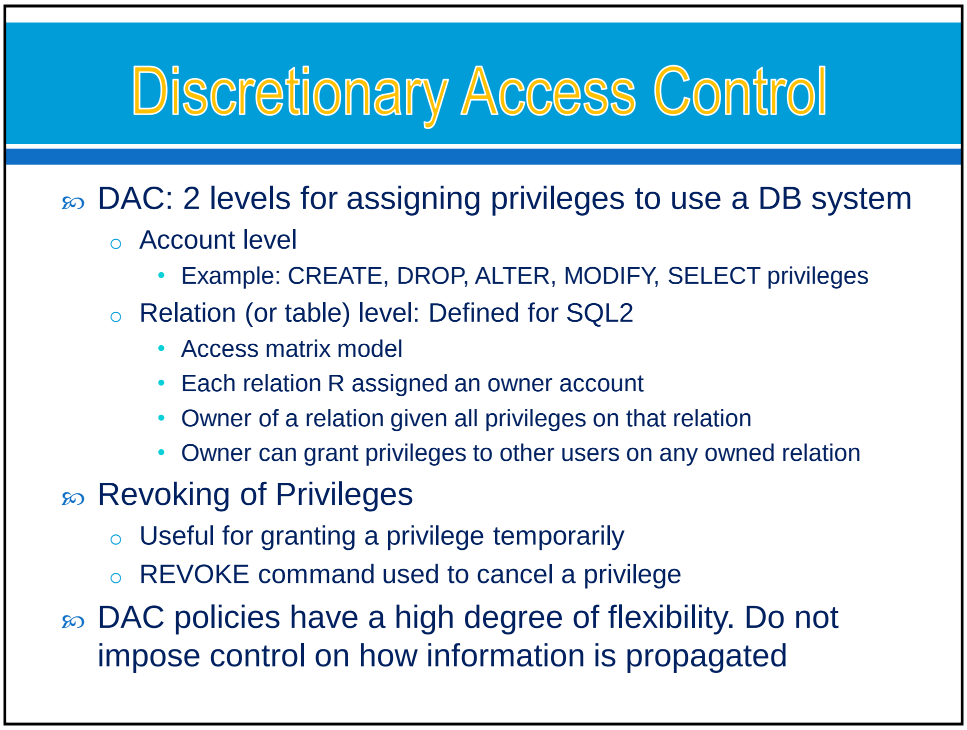
15/10/2024 17



Database Security and the DBA

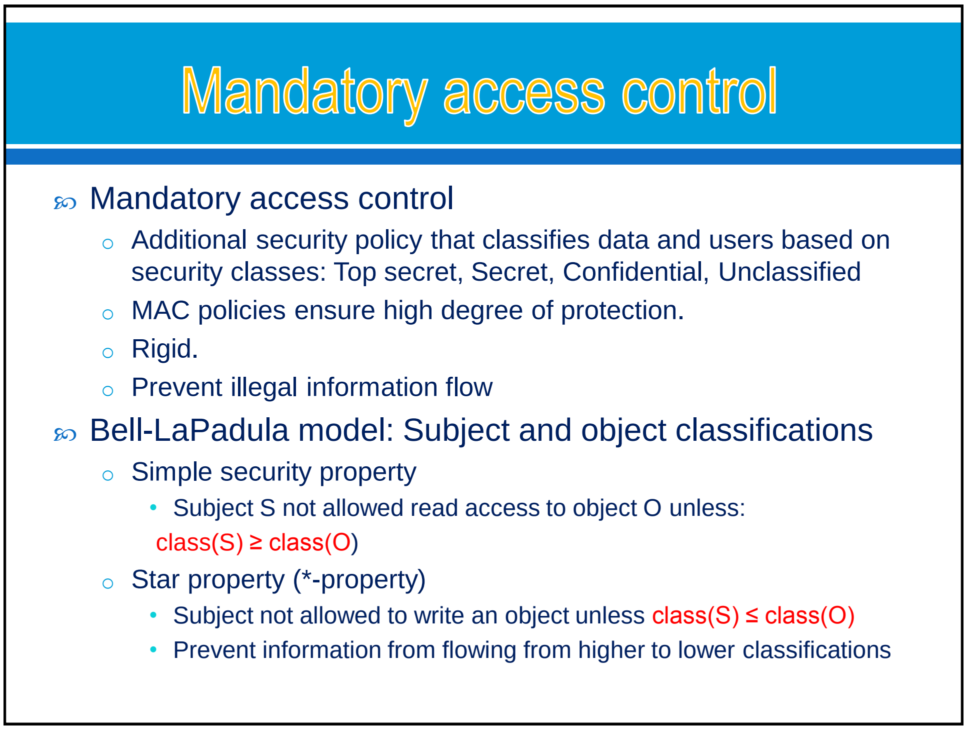
- Database administrator (DBA)
  - Central authority for administering database system
  - Superuser or system account
- The DBA is responsible for the overall security of the database system.
  - granting privileges to users who need to use the system
  - classifying users and data in accordance with the policy of the organization
- DBA-privileged commands
  - Account creation - access control
  - Privilege granting - control discretionary authorization
  - Privilege revocation - control discretionary authorization
  - Security level assignment - control mandatory authorization

Slide 30- 18



Discretionary Access Control

- DAC: 2 levels for assigning privileges to use a DB system
  - Account level
    - Example: CREATE, DROP, ALTER, MODIFY, SELECT privileges
  - Relation (or table) level: Defined for SQL2
    - Access matrix model
    - Each relation R assigned an owner account
    - Owner of a relation given all privileges on that relation
    - Owner can grant privileges to other users on any owned relation
- Revoking of Privileges
  - Useful for granting a privilege temporarily
  - REVOKE command used to cancel a privilege
- DAC policies have a high degree of flexibility. Do not impose control on how information is propagated

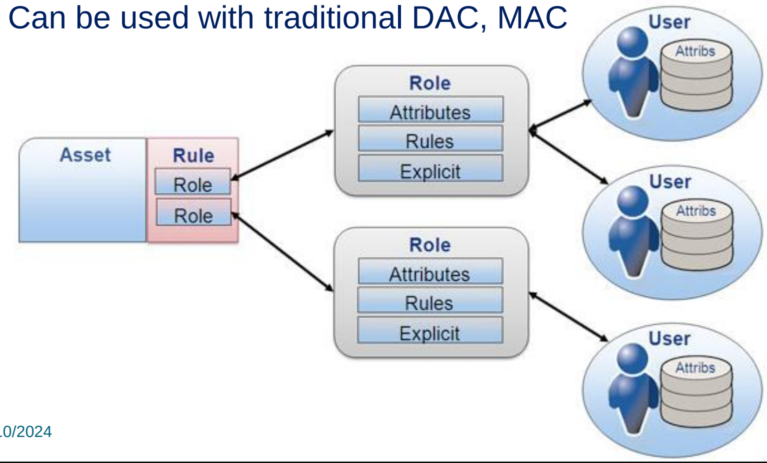


Mandatory access control

- Mandatory access control
  - Additional security policy that classifies data and users based on security classes: Top secret, Secret, Confidential, Unclassified
  - MAC policies ensure high degree of protection.
  - Rigid.
  - Prevent illegal information flow
- Bell-LaPadula model: Subject and object classifications
  - Simple security property
    - Subject S not allowed read access to object O unless: class(S) ≥ class(O)
  - Star property (*-property)
    - Subject not allowed to write an object unless class(S) ≤ class(O)
    - Prevent information from flowing from higher to lower classifications

Role-based Access control

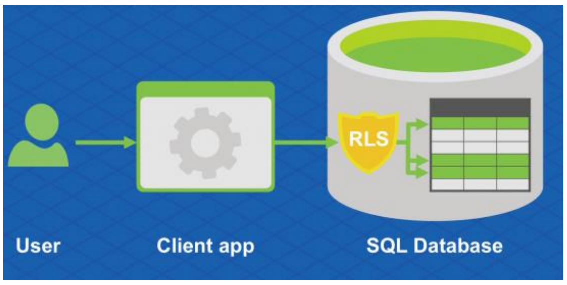
- Permissions associated with organizational roles
  - Users are assigned to appropriate roles
- Can be used with traditional DAC, MAC



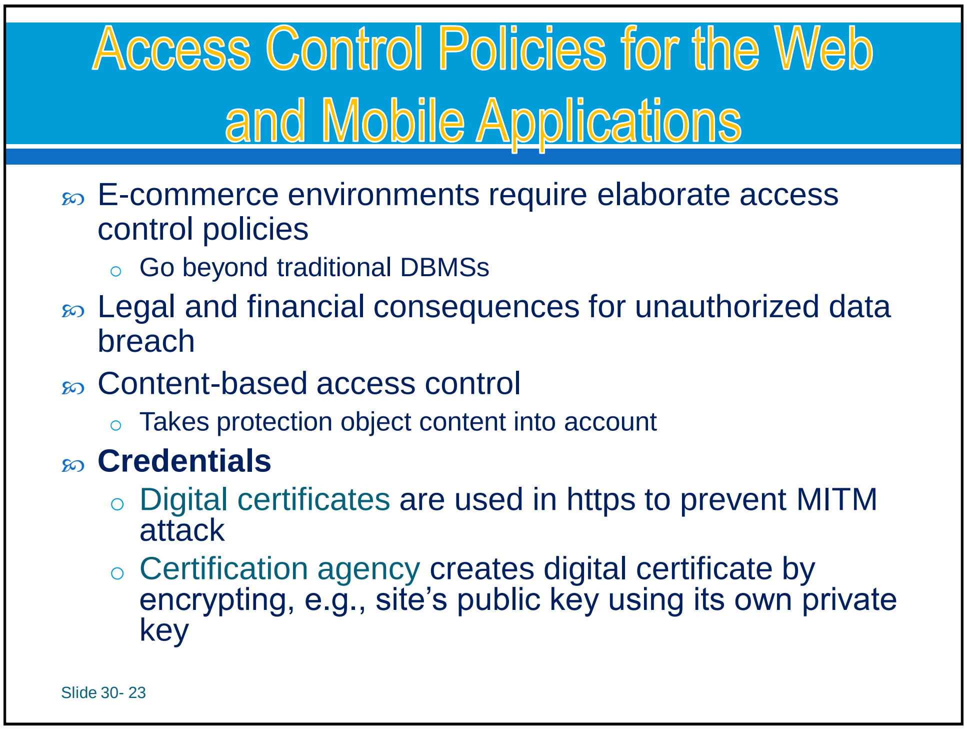
15/10/2024 21

Row-Level Access Control

- Sophisticated access control rules implemented by considering the data row by row
- Each row given a label
  - prevent unauthorized users from viewing or altering certain data
- Provides finer granularity of data security
- Label security policy: Defined by an administrator
- On top of DAC (must satisfy DAC and the label security requirements)



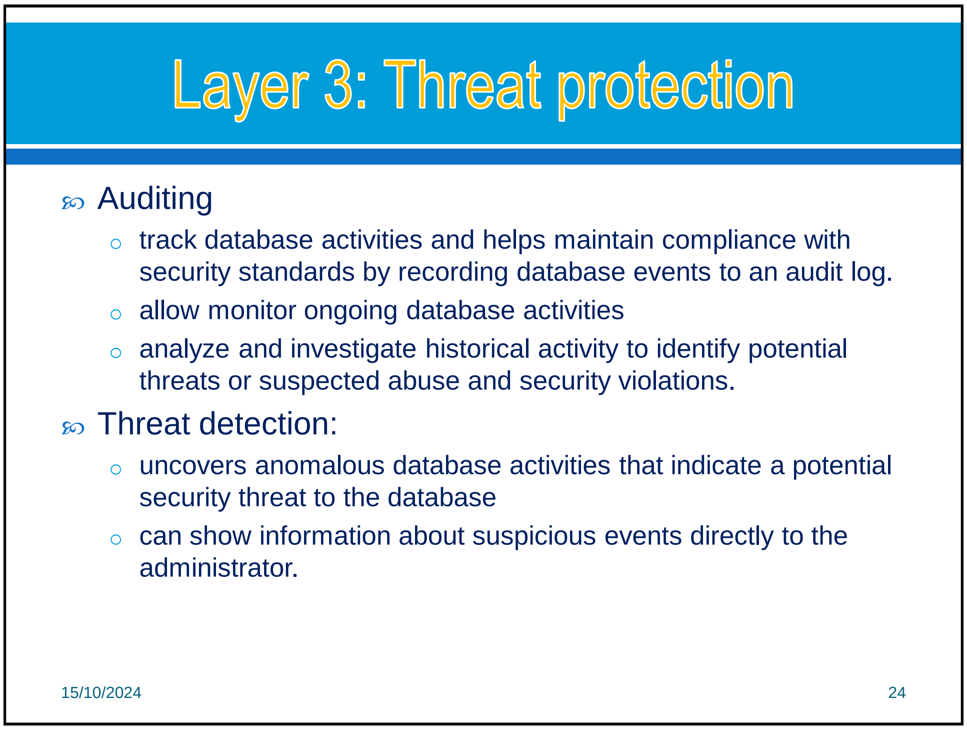
Slide 30- 22



Access Control Policies for the Web and Mobile Applications

- E-commerce environments require elaborate access control policies
  - Go beyond traditional DBMSs
- Legal and financial consequences for unauthorized data breach
- Content-based access control
  - Takes protection object content into account
- **Credentials**
  - Digital certificates are used in https to prevent MITM attack
  - Certification agency creates digital certificate by encrypting, e.g., site’s public key using its own private key

Slide 30- 23



Layer 3: Threat protection

- Auditing
  - track database activities and helps maintain compliance with security standards by recording database events to an audit log.
  - allow monitor ongoing database activities
  - analyze and investigate historical activity to identify potential threats or suspected abuse and security violations.
- Threat detection:
  - uncovers anomalous database activities that indicate a potential security threat to the database
  - can show information about suspicious events directly to the administrator.

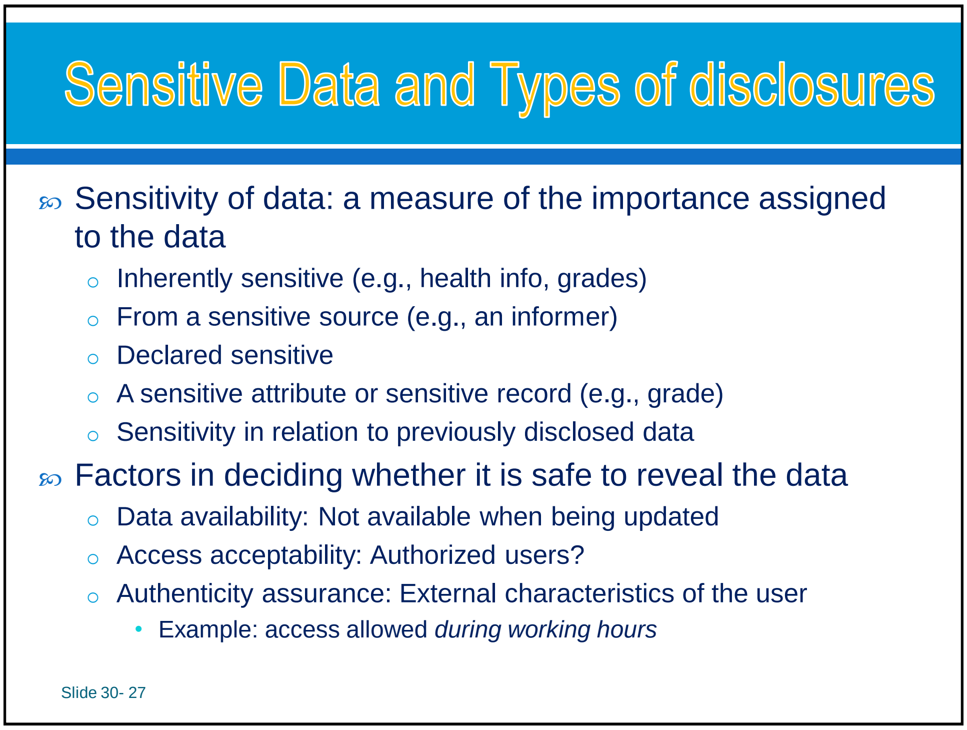
15/10/2024 24

Layer 4: Information protection

- **Data encryption**
  - it enhances security by limiting data loss when access controls are bypassed.
- **Database backup data and recovery**
  - involves making backup copies of the database and log files on a regular basis and storing the copies in a secure location.
  - They are available to restore the database in the event of a security breach or failure.
- **Security for web server applications and websites:**
  - Any application or web server that connects to the database could be a target and should be subjected to periodic security testing and best practices management.
- **Physical security**
  - limits access to the physical server and hardware components.

Comprehensive data encryption

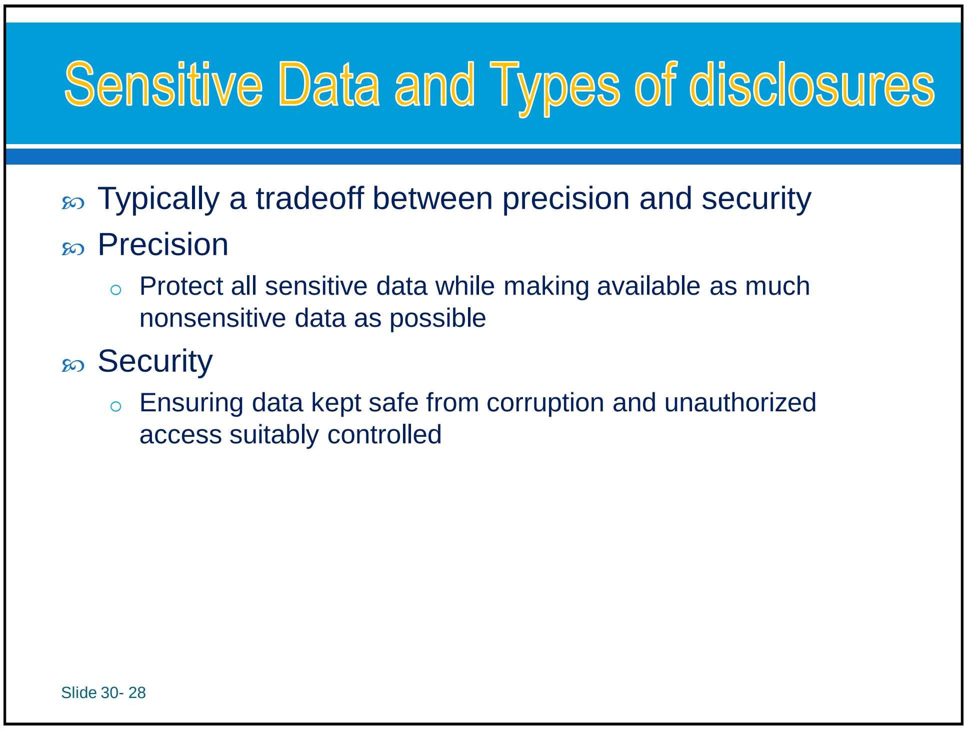
- Always encrypted data offers built-in protection of data against theft in memory, on disk, in transit, and even during query processing.
  - Encrypt the database at storage level, transparent to application
    - Whole database/file/relation: Unit of encryption: page
    - Column encryption
    - Supported by many database systems
  - Encrypt "in transit" between the client and server irrespective - Network level – using TLS protocol: use encryption to prevent
    - Eavesdropping: unauthorized reading of messages
    - Masquerading: pretending to be an authorized user



Sensitive Data and Types of disclosures

- Sensitivity of data: a measure of the importance assigned to the data
  - Inherently sensitive (e.g., health info, grades)
  - From a sensitive source (e.g., an informer)
  - Declared sensitive
  - A sensitive attribute or sensitive record (e.g., grade)
  - Sensitivity in relation to previously disclosed data
- Factors in deciding whether it is safe to reveal the data
  - Data availability: Not available when being updated
  - Access acceptability: Authorized users?
  - Authenticity assurance: External characteristics of the user
    - Example: access allowed *during working hours*

Slide 30- 27



Sensitive Data and Types of disclosures

- Typically a tradeoff between precision and security
- Precision
  - Protect all sensitive data while making available as much nonsensitive data as possible
- Security
  - Ensuring data kept safe from corruption and unauthorized access suitably controlled

Slide 30- 28

Relationship Between Information Security and Information Privacy

- Security: technology to ensure info protection
- Concept of privacy goes beyond security
  - Ability of individuals to control the terms under which their personal information is acquired and used
  - Preventing storage of personal information
  - Ensuring appropriate use of personal information
- Security a required building block for privacy

Slide 30- 29

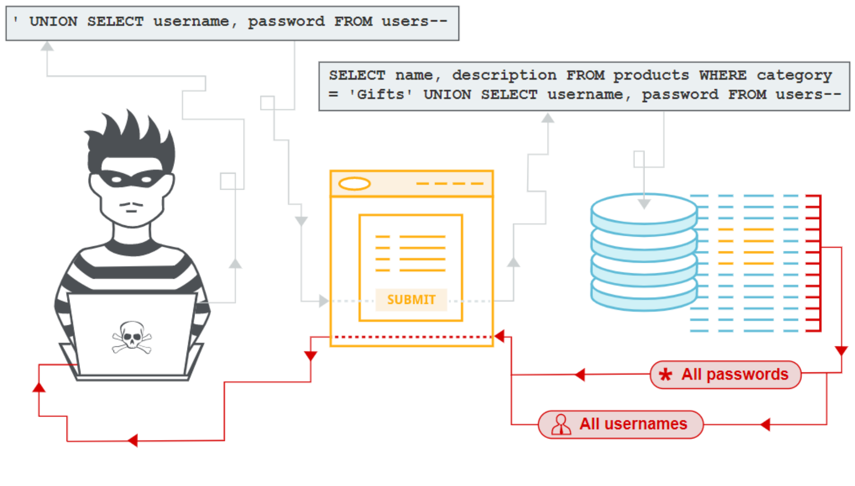
Attacks on Databases

- SQL Injection
- Inference attacks

SQL Injections

- Malicious SQL commands are sent to a database
- Can impact both
  - confidentiality (extraction of data) and
  - integrity (corruption of data)
- In a web application environment, typically a script takes user input and builds an SQL query
- Web application vulnerability can be used to craft an SQL injection
- SQL injection attack is one of the most prevalent and dangerous network-based security threats

SQL Injection



```
' UNION SELECT username, password FROM users--
```

```
SELECT name, description FROM products WHERE category
= 'Gifts' UNION SELECT username, password FROM users--
```

SQL Injection Example

The SQLi attack typically works:

- early terminating a text string
- appending a new command.
- terminates the injected string with a comment mark "--".

Example:

```
Var Shipcity;
Shipcity = Request.form ("Shipcity");
Var sql = "select * from OrdersTable
where
Shipcity = '" + Shipcity + "'";
```

a user will enter the name of a city. Ex, REDMOND,

- Script generates:

SELECT * FROM OrdersTable Where Shipcity = 'Redmond'.

SQL Injection Example

- What if user enters:

Redmond' ; DROP table OrdersTable--

- In this case, script is generated:

SELECT * FROM OrdersTable WHERE Shipcity = 'Redmond' ; DROP OrdersTable

⇒ Server will:

- select all records in OrdersTable where ShipCity is Redmond.
- Then, it executes the DROP request

- Malicious user is able to inject code to delete the table
- Many other code injection examples exist

SQLi Attack Avenues

- **User input**: In this case, attackers inject SQL commands by providing suitably crafted user input.
- **Server variables**: variables are logged to a database without sanitization, this could create an SQL injection vulnerability.
- **Second-order injection**: a malicious user could rely on data already present in the system or database to trigger an SQL injection attack
- **Cookies:** an attacker could alter cookies when the application server builds an SQL query based on the cookie’s content, the structure and function of the query is modified.
- **Physical user input**: could be scanned using optical character recognition and passed to a database management system.

The impact of a SQL injection attack

- A successful SQL injection attack can result in unauthorized access to sensitive data, such as passwords, credit card details, or personal user information.
- Many high-profile data breaches, leading to reputational damage and regulatory fines.
- In some cases, an attacker can obtain a persistent backdoor into an organization's systems,
  - leading to a long-term compromise that can go unnoticed for an extended period.

SQL injection types

- There are a wide variety of SQL injection vulnerabilities, attacks, and techniques, which arise in different situations. Some common SQL injection examples include:
  - Retrieving hidden data, where you can modify an SQL query to return additional results.
  - Subverting application logic, where you can change a query to interfere with the application's logic.
  - UNION attacks, where you can retrieve data from different database tables.
  - Examining the database, where you can extract information about the version and structure of the database.
  - Blind SQL injection, where the results of a query you control are not returned in the application's responses.

Retrieving hidden data

- When the user clicks on the Gifts category, the URL:
  - https://insecure-website.com/products?category=Gifts
  - An SQL query to retrieve details:

  SELECT * FROM products WHERE category = 'Gifts' AND released = 1
- Attaker attacker can construct an attack like:
  - **Ex1:** https://insecure-website.com/products?category=Gifts'--
  - This results in the SQL query:

  SELECT * FROM products WHERE category = 'Gifts'--' AND released = 1

  -- is a comment: removes the remainder of the query, so it no longer includes AND released = 1. This means that all products are displayed
  - **Ex2:** https://insecure-website.com/products?category=Gifts'+OR+1=1--
  - This results in the SQL query:

  SELECT * FROM products WHERE category = 'Gifts' OR 1=1--' AND released = 1
  - The modified query will return all items where either the category is Gifts, or 1 is equal to 1. Since 1=1 is always true, the query will return all items.

Subverting application logic

- If a user submits the username wiener and the password bluecheese, the application checks the credentials by performing the following SQL query:
  - SELECT * FROM users WHERE username = 'wiener' AND password = 'bluecheese'
  - If the query returns the details of a user, then the login is successful. Otherwise, it is rejected.
- Here, an attacker can log in as any user without a password simply by using the SQL comment sequence -- to remove the password check from the WHERE clause of the query.
  - SELECT * FROM users WHERE username = 'administrator'--' AND password = ''
  - This query returns the user whose username is administrator and successfully logs the attacker in as that user.

Retrieving data from other db tables

- In cases where the results of an SQL query are returned within the application's responses, an attacker can leverage an SQL injection vulnerability to retrieve data from other tables within the database.
- This is done using the UNION keyword, which lets you execute an additional SELECT query and append the results to the original query.
- For example, if an application executes the following query containing the user input "Gifts":
  - SELECT name, description FROM products WHERE category = 'Gifts'

  then an attacker can submit the input:
  - ' UNION SELECT username, password FROM users--

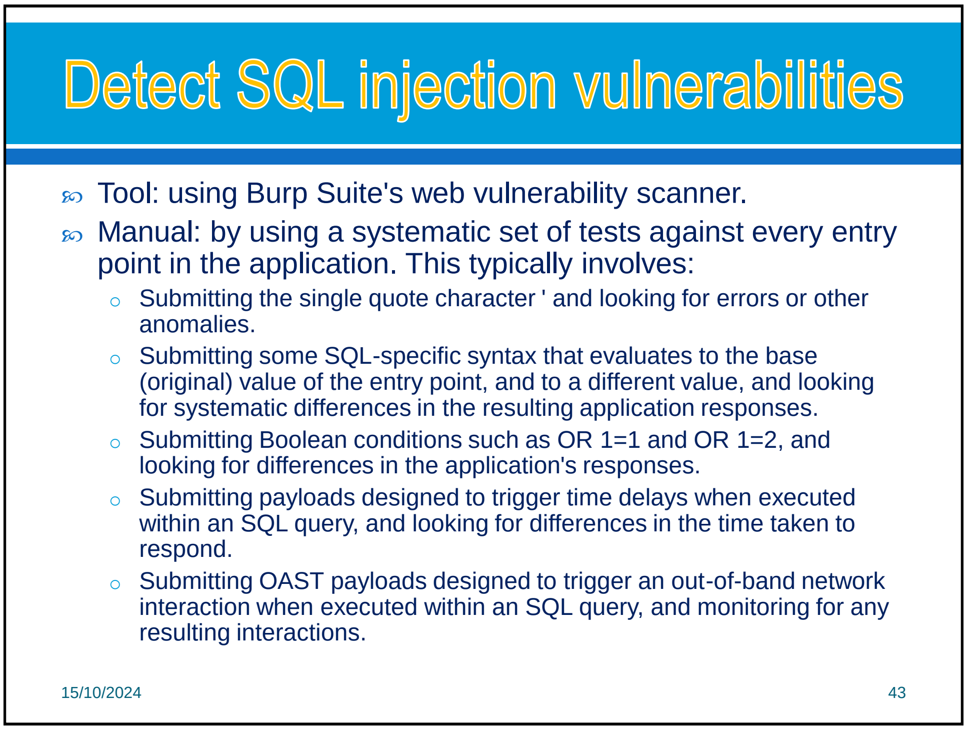
  This will cause the application to return all usernames and passwords along with the names and descriptions of products.

Examining the database

- Following initial identification of an SQL injection vulnerability, it is generally useful to obtain some information about the database itself. This information can often pave the way for further exploitation.
- You can query the version details for the database. The way that this is done depends on the database type, so you can infer the database type from whichever technique works. For example, on Oracle you can execute:
  - SELECT * FROM v$version
- You can also determine what database tables exist, and which columns they contain. For example, on most databases you can execute the following query to list the tables:
  - SELECT * FROM information_schema.tables

Blind SQL injection vulnerabilities

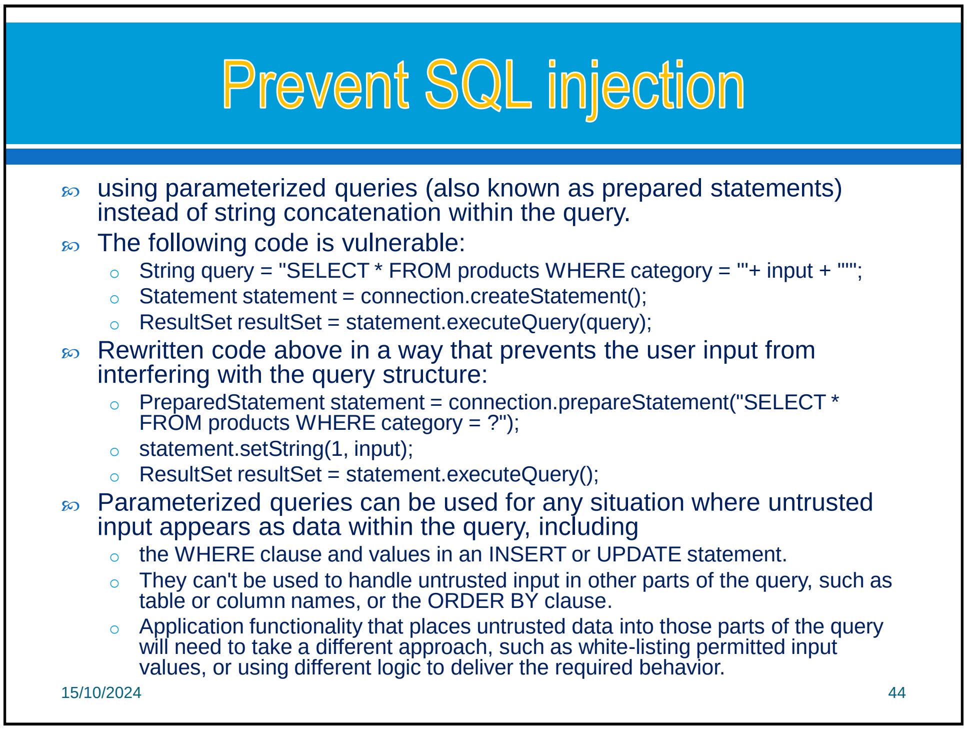
- Blind SQL injection: Many instances of SQL injection are blind vulnerabilities.
  - does not return the results of the SQL query or the details of any database errors within its responses.
  - can still be exploited to access unauthorized data, but the techniques involved are generally more complicated and difficult to perform.
- Techniques can be used to exploit blind SQL injection vulnerabilities:
  - change the logic of the query to trigger a detectable difference in the application's response depending on the truth of a single condition. This might involve injecting a new condition into some Boolean logic, or conditionally triggering an error such as a divide-by-zero.
  - trigger a time delay in the processing of the query, allowing you to infer the truth of the condition based on the time that the application takes to respond.
  - can trigger an out-of-band network interaction, using OAST techniques. This technique is extremely powerful and works in situations where the other techniques do not. Often, you can directly exfiltrate data via the out-of-band channel, for example by placing the data into a DNS lookup for a domain that you control.



Detect SQL injection vulnerabilities

- Tool: using Burp Suite's web vulnerability scanner.
- Manual: by using a systematic set of tests against every entry point in the application. This typically involves:
  - Submitting the single quote character ' and looking for errors or other anomalies.
  - Submitting some SQL-specific syntax that evaluates to the base (original) value of the entry point, and to a different value, and looking for systematic differences in the resulting application responses.
  - Submitting Boolean conditions such as OR 1=1 and OR 1=2, and looking for differences in the application's responses.
  - Submitting payloads designed to trigger time delays when executed within an SQL query, and looking for differences in the time taken to respond.
  - Submitting OAST payloads designed to trigger an out-of-band network interaction when executed within an SQL query, and monitoring for any resulting interactions.

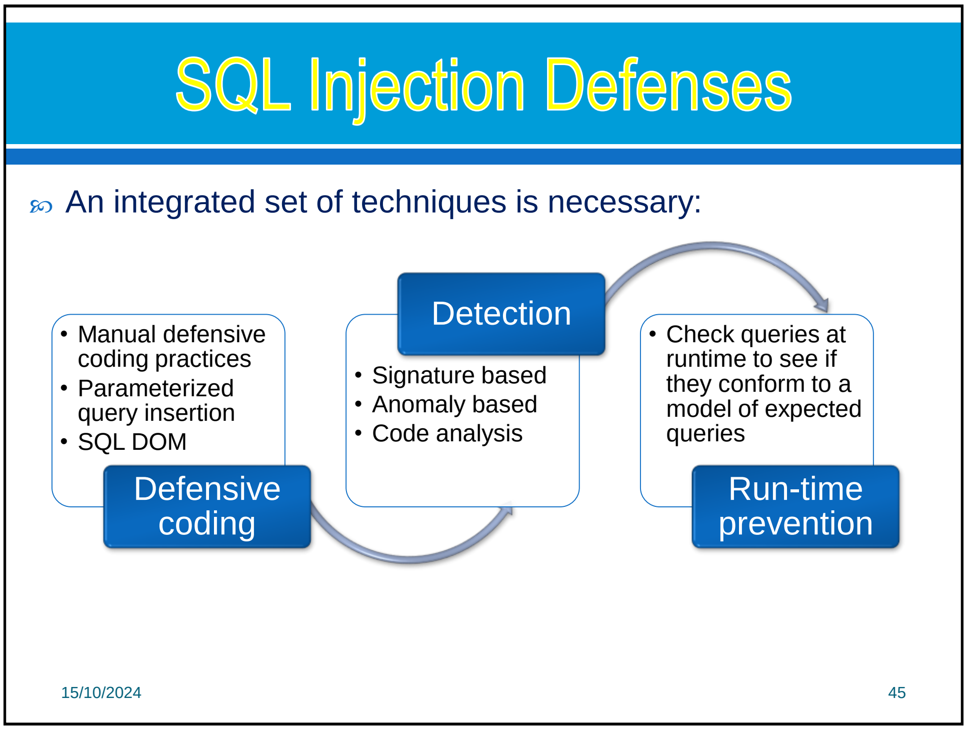
15/10/2024 43



Prevent SQL injection

- using parameterized queries (also known as prepared statements) instead of string concatenation within the query.
- The following code is vulnerable:
  - String query = "SELECT * FROM products WHERE category = '"+ input + "'";
  - Statement statement = connection.createStatement();
  - ResultSet resultSet = statement.executeQuery(query);
- Rewritten code above in a way that prevents the user input from interfering with the query structure:
  - PreparedStatement statement = connection.prepareStatement("SELECT * FROM products WHERE category = ?");
  - statement.setString(1, input);
  - ResultSet resultSet = statement.executeQuery();
- Parameterized queries can be used for any situation where untrusted input appears as data within the query, including
  - the WHERE clause and values in an INSERT or UPDATE statement.
  - They can't be used to handle untrusted input in other parts of the query, such as table or column names, or the ORDER BY clause.
  - Application functionality that places untrusted data into those parts of the query will need to take a different approach, such as white-listing permitted input values, or using different logic to deliver the required behavior.

15/10/2024 44



SQL Injection Defenses

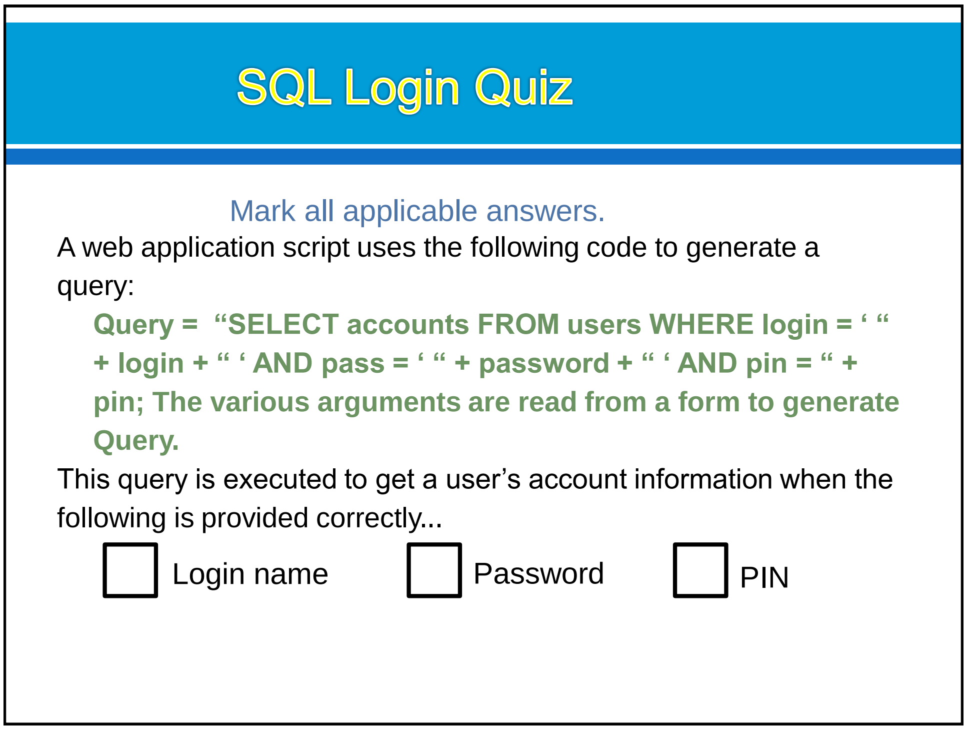
- An integrated set of techniques is necessary:

Defensive coding
- Manual defensive coding practices
- Parameterized query insertion
- SQL DOM

Detection
- Signature based
- Anomaly based
- Code analysis

Run-time prevention
- Check queries at runtime to see if they conform to a model of expected queries

15/10/2024 45



SQL Login Quiz

Mark all applicable answers.

A web application script uses the following code to generate a query:

Query = “SELECT accounts FROM users WHERE login = ‘ “ + login + “ ‘ AND pass = ‘ “ + password + “ ‘ AND pin = “ + pin; The various arguments are read from a form to generate Query.

This query is executed to get a user’s account information when the following is provided correctly...

☐ Login name ☐ Password ☐ PIN

SQL Login Quiz #2

Choose the best answer.

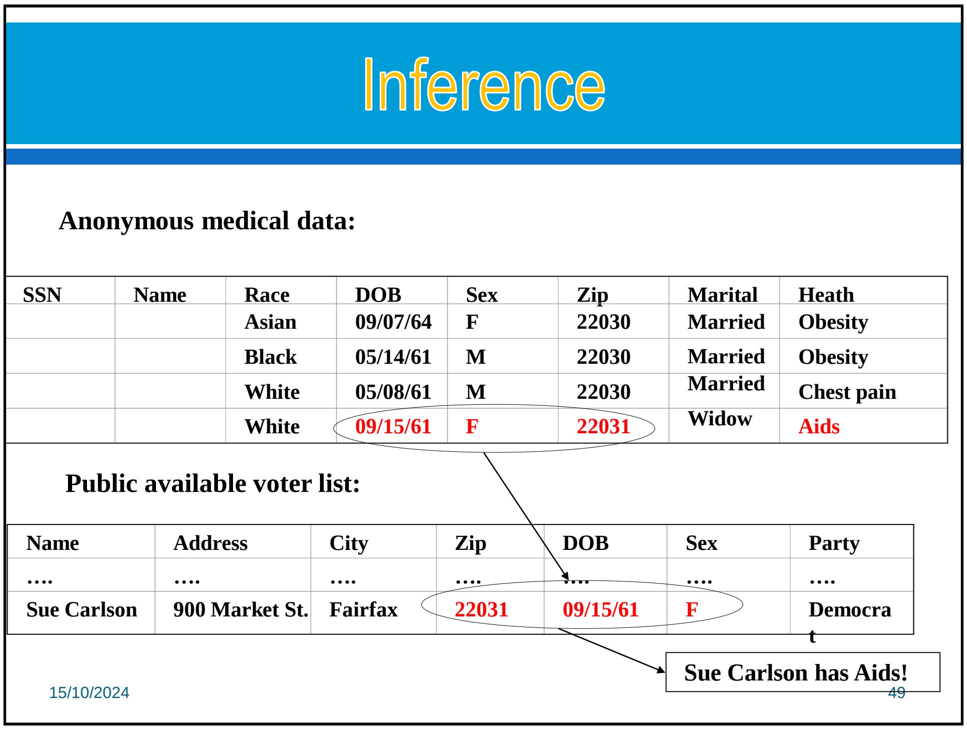
Query = "SELECT accounts FROM users WHERE login = ' " + login + " ' AND pass = ' " + password + " ' AND pin = " + pin; The various arguments are read from a form to generate Query.

If a user types "'or 1 = 1 --" for login in the above query...

- ☐ Query will fail because the provided login is not a correct user
- ☐ An injection attack will result in all users' account data being returned

Inference Attacks on Databases

- Inference attacks:
  - relates to database security
  - is the process of performing authorized queries and deducing unauthorized information from the legitimate responses received.
- Problem:
  - the combination of a number of data items is more sensitive than the individual items,
  - the combination of data items can be used to infer data of a higher sensitivity



Inference

Anonymous medical data:

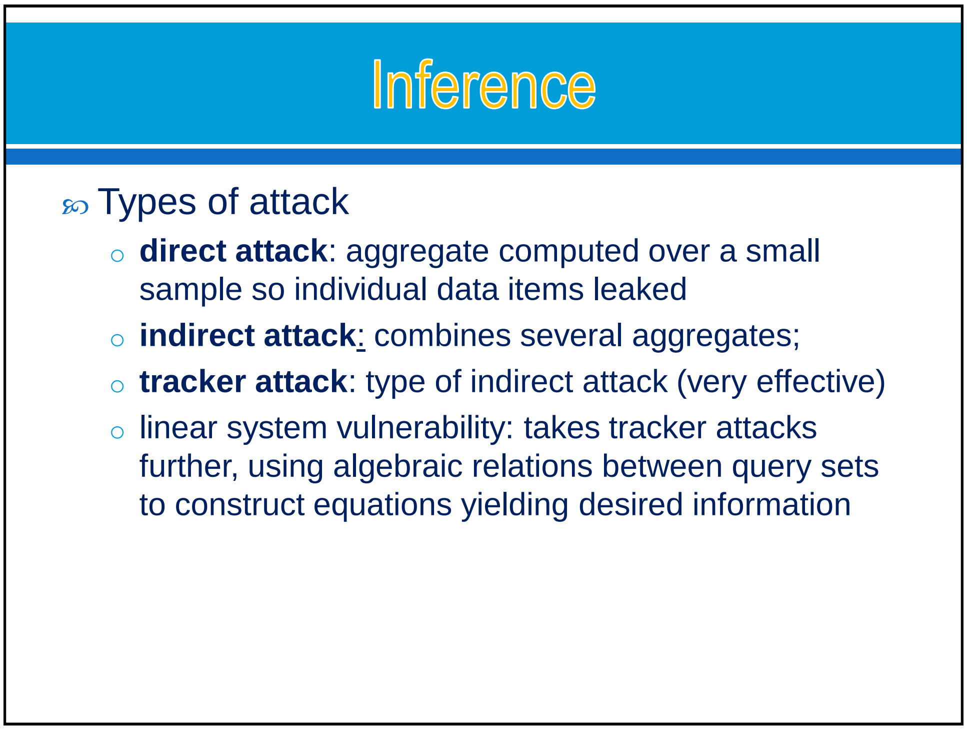
SSN	Name	Race	DOB	Sex	Zip	Marital	Heath
		Asian	09/07/64	F	22030	Married	Obesity
		Black	05/14/61	M	22030	Married	Obesity
		White	05/08/61	M	22030	Married	Chest pain
		White	09/15/61	F	22031	Widow	Aids

Public available voter list:

Name	Address	City	Zip	DOB	Sex	Party
….	….	….	….	….	….	….
Sue Carlson	900 Market St.	Fairfax	22031	09/15/61	F	Democrat

Sue Carlson has Aids!

15/10/2024 49



Inference

- Types of attack
  - **direct attack**: aggregate computed over a small sample so individual data items leaked
  - **indirect attack**: combines several aggregates;
  - **tracker attack**: type of indirect attack (very effective)
  - linear system vulnerability: takes tracker attacks further, using algebraic relations between query sets to construct equations yielding desired information

Inference, ex

NAME	SEX	RACE	AID	FINES	DRUGS	DORM
Adams	M	C	5000	45	1	Holmes
Bailey	M	B	0	0	0	Grey
Chin	F	A	3000	20	0	West
Dewitt	M	B	1000	35	3	Grey
Earhart	F	C	2000	95	1	Holmes
Fein	F	C	1000	15	0	West
Groff	M	C	4000	0	3	West
Hill	F	B	5000	10	2	Holmes
Koch	F	C	0	0	1	West
Liu	F	A	0	10	2	Grey
Majors	M	C	2000	0	2	Grey

Inference - Direct Attack

- Direct Attack
  - determine values of sensitive fields by seeking them directly with queries that yield few records
  - request LIST which is a union of 3 sets

    LIST NAME where (SEX =M ∧ DRUGS = 1) ∨

    (SEX ≠ M ∧ SEX ≠F) ∨ (DORM = Ayres)
    - No dorm named Ayres , Sex either M or F
  - “*n* items over *k* percent” rule helps prevent attack

Inference - Indirect attack

Indirect attack: combines several aggregates

Sums of Financial Aid by Dorm and Sex

	Holmes	Grey	West	Total
M	5000	3000	4000	12000
F	7000	0	4000	11000
Total	12000	3000	8000	23000

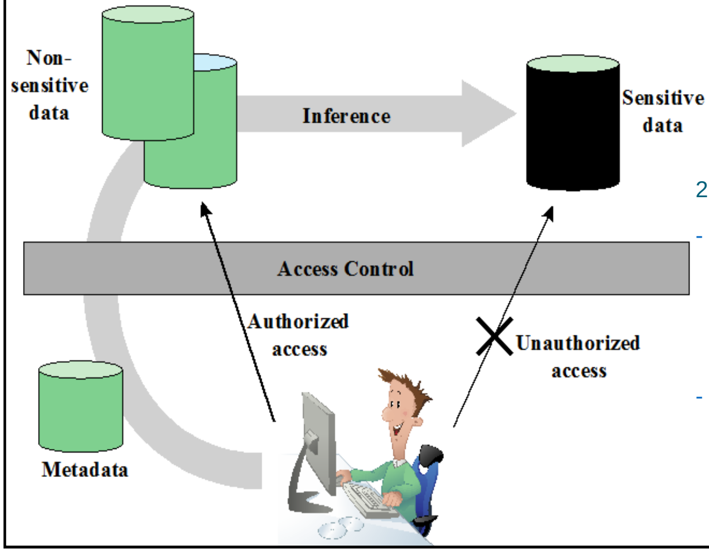
Students by Dorm and Sex

	Holmes	Grey	West	Total
M	1	3	1	5
F	2	1	3	6
Total	3	4	4	11

- 1 Male in Holmes receives 5000
- 1 Female in Grey received no aid

- request a list of names by dorm (non sensitive)

Indirect Information Access via Inference Channel



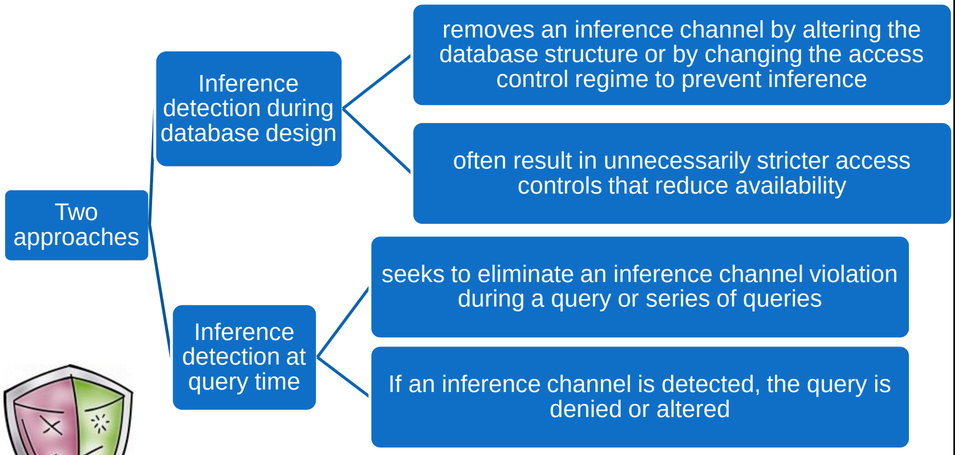
2 inference techniques:

- analyzing functional dependencies between attributes within a table or across tables,
- merging views with the same constraints.

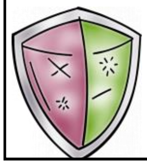
Inference - tracker attack

- Often databases protected against delivering small response sets to queries
- Trackers can identify unique value
  - request (n) and (n-1) values
  - given *n* and *n* – 1, we can easily compute the desired single element

Defenses Against Inference Attacks



- Two approaches
  - Inference detection during database design
    - removes an inference channel by altering the database structure or by changing the access control regime to prevent inference
    - often result in unnecessarily stricter access controls that reduce availability
  - Inference detection at query time
    - seeks to eliminate an inference channel violation during a query or series of queries
    - If an inference channel is detected, the query is denied or altered



SQL Inference Attack Quiz

Choose the best answer.

The database that stores student exam scores allows queries that return average score for students coming from various states. Can this lead to an inference attack in this system?

- ☐ Yes, depending on how many students come from each state
- ☐ No, it is not possible

SQL Inference Attack Quiz #2

Choose the best answer.

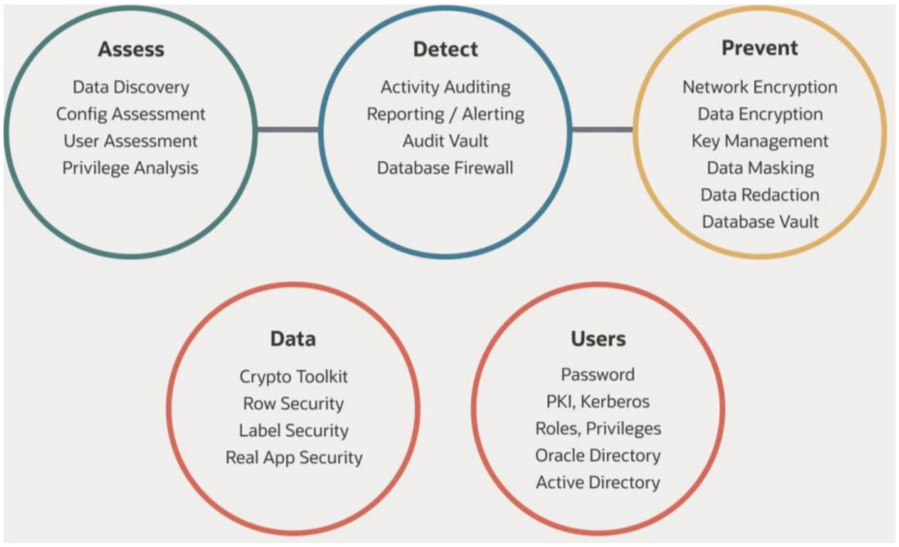
Assume in (1), the data in the database is de-identified by removing student id (and other information such as names). Furthermore, the field that has the state of the student is generalized by replacing it with the US region (e.g., Midwest). The generalization ensures that there are at least two students from each region. Are inference attacks still possible?

☐ Yes ☐ No

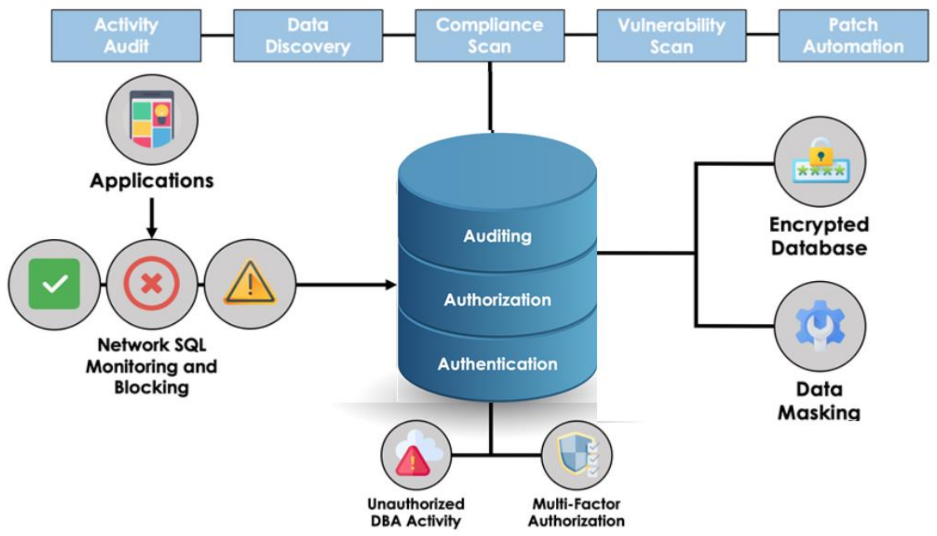
Database encryption

- The database is protected by multiple layers of security:
  - Firewalls
  - Authentication mechanisms
  - General access control systems
  - Database access control systems.
- Database encryption is warranted and often implemented for particularly sensitive data
- There are two disadvantages to database encryption:
  • **Key management:** Authorized users must have access to the decryption key for the data. Providing secure keys to selected parts of the database to authorized users and applications is a complex task.
  • **Inflexibility:** When part or all of the database is encrypted, it becomes more difficult to perform record searching.

Database security best practices



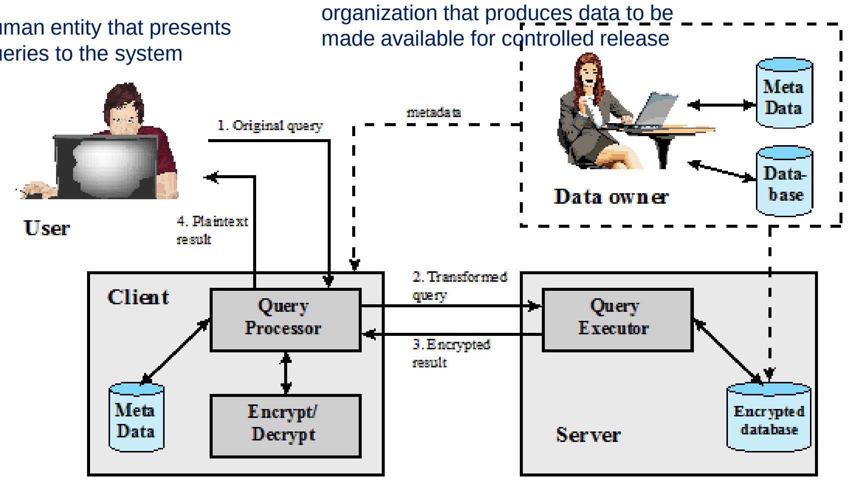
Oracle's Maximum Database Security Architecture



A Database Encryption Scheme

human entity that presents queries to the system

organization that produces data to be made available for controlled release



frontend that transforms user queries into queries on the encrypted data stored on the server

an organization that receives the encrypted data from a data owner and makes them available for distribution to clients

Summary

- Used to **store lots of sensitive data** that can be accessed via programs (queries)
- Access control must be **based on operations allowed by databases**
- New attacks on databases arise due to their unique characteristics
- Defenses **must address such attacks**

Lab: Database Security

1. DVWA
   - Get important information in DVWA database such as: tables, user/pass with different level: Low, Medium, High
2. Sqlmap:
   - Get important information in DVWA database: tables, user/pass with different level: Low, Medium, High
   - Database from other website, ex:
     - http://testphp.vulnweb.com
3. Other Tools:
   - Hackbar (built-in web browser) -> vulnerable website.